1. Doominojada

Seda ülesannet on kõige lihtsam lahendada nii, et teha maksimaalne (28 kivist koosnev) jada käsitsi valmis, kirjutada see programmi teksti sisse ja siis töö ajal väljastada selle algusest vajaliku pikkusega jupp. Nii ongi tehtud failis `djlah1.pas`.

Võib lahendada ka variantide läbivaatusega. Esmapilgul tundub küll, et 28 kivi järjestamiseks on liiga palju võimalusi (kõigi võimalike ümberjärjestuste läbivaatus mahuks ühe sekundi sisse umbes 10 kivi jaoks), aga tegelikult on ka võimalikke lahendusi väga palju ja praktikas ei jää failis `djlah2.pas` toodud lahendus ajahätta üheski testis.

Variantide läbivaatuse võib programmeerida ka ilma rekursiooni kasutamata, nagu on tehtud failis `djlah3.pas` toodud lahenduses, mis lisaks vastusele väljastab ka statistikat selle kohta, kui palju lahenduse leidmiseks tööd tehti.

Testid

1. $T = 1$. 2 punkti.
2. $T = 4$. 2 punkti.
3. $T = 7$. 2 punkti.
4. $T = 10$. 2 punkti.
5. $T = 13$. 2 punkti.
6. $T = 16$. 2 punkti.
7. $T = 19$. 2 punkti.
8. $T = 22$. 2 punkti.
9. $T = 25$. 2 punkti.
10. $T = 28$. 2 punkti.

Kokku 20 punkti.

2. Telemastid

Selle ülesande lahendamiseks võib proovida igale mastile kõiki võimalikke asukohti ja arvutada iga paigutuse jaoks, milline peab olema saatjate tegevusraadius, et kõik linnad oleks leviga kaetud. Vajaliku raadiuse leidmiseks võib arvutada iga linna jaoks tema kauguse kõigist kolmest mastist. Neist kolmest kaugusest minimaalne ongi tegevusraadius, mis on vaja, et see linn oleks levis. Kõigi linnade vajalikest raadiustest omakorda maksimaalne on aga see, mis määrab saatjate hinna ja seda tulekski minimeerida.

Sellise lahenduse naiivne realisatsioon kaaluks iga masti jaoks umbes 400 võimalikku asukohta ja kulutaks N linna korral kokku umbes $400^3N = 64\,000\,000N$ operatsiooni, mis mahuks ajapiiridesse vaid $N = 1$ ja $N = 2$ korral.

Ajakulu saab kokku hoida, kui paneme tähele, et mastide omavahel vahetamine vajalikku saateraadiust ei mõjuta. Seega võime mastide paigutusvariante läbi vaadates kaaluda näiteks ainult selliseid, kus esimene mast ei ole teisest ja teine kolmandast lõuna pool. Selline lahendus on toodud failis `tvlah1.pas` ja mahub suurimates testides napilt ajalimiidi sisse, kulutades N linnaga testis umbes $10\,000\,000N$ operatsiooni.

Veel mõned võimalused aja kokkuhoiuks:

- Kunagi ei ole kasulik panna masti kõige põhjapoolsemast linnast põhja poole, kõige lõunapoolsemast linnast lõuna poole jne. Sellega on võimalik paljudes testides oluliselt vähendada iga masti võimalike asukohtade arvu ja sellega ka võimalike paigutuste koguarvu.
- Kui mingi paigutuse hindamisel selgub, et ühe linna kaugus kõigist mastidest on suurem kui seni parimas lahenduses vajalik saateraadius, siis pole mõtet seda paigutust teiste linnade jaoks enam edasi uurida.

Neid võimalusi kasutabki ära failis `tvlah2.cpp` toodud lahendus.

Testid

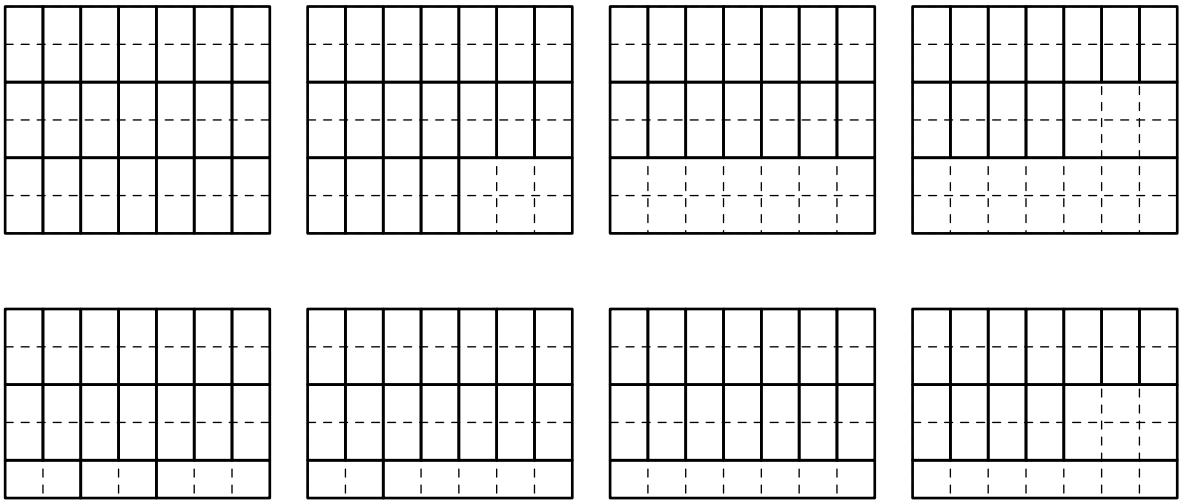
1. $N = 3$. Minimaalne test. 4 punkti.
2. $N = 8$. 4 punkti.
3. $N = 8$. 4 punkti.
4. $N = 4$. Linnad kaardi nurkades, üksteisest maksimaalselt kaugel. 4 punkti.
5. $N = 10$. Maksimaalne test. 4 punkti.
6. $N = 10$. Maksimaalne test. Linnad kõik ühel sirgel. 4 punkti.
7. $N = 9$. Linnad moodustavad 3×3 ruudustiku. 4 punkti.
8. $N = 10$. Maksimaalne test. Linnad asuvad sümmeetriliselt kaardi servadel. 4 punkti.
9. $N = 6$. 4 punkti.
10. $N = 7$. 4 punkti.

Kokku 40 punkti.

3. Ristküliku tükeldamine

Üks võimalik lahendus: hakkame ristkülikut ridade kaupa 2×1 tükkidega täitma.

Siis on võimalikud järgmised erijuhud:



Failis `rtlah1.pas` toodud lahendus käistleb neid kõiki üsna kompaktselt.

Testid

Paaris N, paaris M

1. $N = 4$, $M = 4$, $K = 2$. 2/1 punkti.
2. $N = 4$, $M = 4$, $K = 4$. 2/1 punkti.
3. $N = 4$, $M = 4$, $K = 8$. Maksimaalne tükkide arv. 2/1 punkti.
4. $N = 10$, $M = 10$, $K = 22$. Keskmise suurusega juhuslik test. 2/1 punkti..
5. $N = 100$, $M = 100$, $K = 4998$. Maksimumilähedane test. 2/1 punkti.

Paaris N, paaritu M

6. $N = 6$, $M = 1$, $K = 3$. Maksimaalne tükkide arv. 2/1 punkti.
7. $N = 6$, $M = 3$, $K = 4$. 2/1 punkti.
8. $N = 6$, $M = 5$, $K = 7$. 2/1 punkti.
9. $N = 16$, $M = 15$, $K = 32$. Keskmise suurusega juhuslik test. 2/1 punkti.
10. $N = 100$, $M = 99$, $K = 4948$. Maksimumilähedane test. 2/1 punkti.

Paaritu N, paaris M

11. $N = 1$, $M = 8$, $K = 3$. 2/1 punkti.
12. $N = 3$, $M = 4$, $K = 6$. Maksimaalne tükkide arv. 2/1 punkti.
13. $N = 5$, $M = 4$, $K = 1$. Minimaalne tükkide arv. 2/1 punkti.
14. $N = 15$, $M = 18$, $K = 40$. Keskmise suurusega juhuslik test. 2/1 punkti.
15. $N = 99$, $M = 100$, $K = 4847$. Maksimumilähedane test. 2/1 punkti.

Paaritu N, paaritu M

16. $N = 3$, $M = 3$, $K = 4$. Maksimaalne tükkide arv. 2/1 punkti.
17. $N = 5$, $M = 5$, $K = 10$. 2/1 punkti.
18. $N = 7$, $M = 9$, $K = 25$. 2/1 punkti.
19. $N = 19$, $M = 13$, $K = 40$. Keskmise suurusega juhuslik test. 2/1 punkti.
20. $N = 99$, $M = 99$, $K = 4897$. Maksimumilähedane test. 2/1 punkti.

Kokku algajatel 40, edasijõudnutel 20 punkti.

4. Bussipiletid

Seda ülesannet tuleks lahendada dünaamilise planeerimise abil.

Üks võimalus on leida iga päeva jaoks odavaim piletite komplekt, millega saaks teha ära kõik sõidud kalendri algusest kuni selle päevani. Päeval, mil sõita pole vaja, on odavaim lahendus ilmselt sama, mis oli odavaim lahendus eelneva päeva jaoks (sest see päev ei pea piletiga kaetud olema).

Päev, mil on vaja sõita, tuleb katta piletiga. Kuna me oleme sel hetkel huvitatud ainult perioodist kalendri algusest kuni selle päeva endani, võime selle päeva valida lisatava pileti kehtivusaja viimaseks päevaks. Sedasi saame läbi proovida kõik erinevad piletid, leides iga pileti jaoks perioodi katmise hinna, liites valitud pileti hinnale selle pileti kehtivusajale eelnenud perioodi katmise hinna. Valides neist variantidest odavaima, saamegi parima lahenduse, kulutades selleks $O(NT)$ operatsiooni.

Ülesandes on veel mõned tehnilised detailid. Kirjeldatud lähenemise puhul (millele vastav lahendus on toodud failis `bplah1.c`) tuleks esimene pilet vahel osta veidi tagasiulatuvalt (enne ajaarvamise algust). Et ajaarvamise algusele eelnevatel päevadel aga meile teadaolevalt sõite ei ole, võib selle pileti sama hästi osta ka esimesel päeval. Analoogiliselt, kui kirjutada lahendus tagurpidi (hakata kalendrit lõpust vaatama, nagu on tehtud failis `bplah1.pas`), võib mõne pileti kehtivusaeg üle kalendri lõpu minna.

Testid

T — piletite arv, N — päevade arv, S — sõidupäevade arv.

1. Lihtne test ühe piletiga: $T = 1$, $N = 70$, $S = 40$. Minimaalne hind 26. 4 punkti.
2. Ilma ühegi sõidupäevata test: $T = 10$, $N = 100$, $S = 0$. Minimaalne hind 0. 4 punkti.
3. Lihtne test 10 piletiga: $T = 10$, $N = 100$, $S = 65$. Minimaalne hind 104. 4 punkti.
4. Test, kus pikim pilet kehtib üle kalendri pikkuse ja seda tuleb kasutada: $T = 12$, $N = 80$, $S = 70$. Minimaalne hind 90. 4 punkti.
5. Test, kus mõned piletid peavad kattuma või kehtima ka peale perioodi lõppu: $T = 13$, $N = 1000$, $S = 560$. Minimaalne hind 829. 4 punkti.
6. Imelike piletitega test (pikema kehtivusega pilet võib olla odavam kui mõni lühem, piletid pole kuidagi järjestatud): $T = 30$, $N = 1000$, $S = 683$. Minimaalne hind 1051. 4 punkti.
7. Ainult sõidupäevadest koosnev test: $T = 17$, $N = 1000$, $S = 1000$. Minimaalne hind 1067. 4 punkti.
8. Tavaline test: $T = 40$, $N = 2000$, $S = 1500$. Minimaalne hind 1973. 4 punkti.
9. Tavaline pikem test: $T = 70$, $N = 5000$, $S = 3200$. Minimaalne hind 3795. 4 punkti.
10. Maksimaalne test: $T = 100$, $N = 10000$, $S = 7000$. Minimaalne hind 4111. 4 punkti.

Kokku 40 punkti.

5. Rändav poliitik

Selles ülesandes üldiselt pole paremat lahendust kui kõigi võimalike sõidujärjekordade läbivaatus, mida on muidugi kõige lihtsam realiseerida rekursiivselt.

Sama linna korduva külastamise vältimiseks on kõige tõhusam hoida iga linna jaoks tõeväärtust, mis näitab, kas seda on jooksvas variandis juba külastatud. Failides `rplah1.c` ja `rplah2.c` toodud lahendused kasutavad selleks täisarvu, milles igale linnale vastab oma bitt: kui see bitt on 1, siis on seda linna juba külastatud. Sama hästi võiks kasutada ka globaalset tõeväärtuste massiivi. Linnade läbimise järjekorra säilitamiseks hoiame läbitud linnade nimekirja ka eraldi massiivis. Selle abil linnade korduva läbimise vältimine oleks aga ebaefektiivne, sest me peaks linna olemasolu tuvastamiseks kogu massiivi läbi käima.

Failis `rplah1.c` toodud lahendus koostab ja sorteerib kasutatud lennupiletite hulga iga marsuudi jaoks eraldi ja jääb paaris suuremas testis ajahätta.

Failis `rplah2.c` toodud lahendus koostab lennupiletite hulga jooksvalt linnade läbimise järjekorra moodustamise ajal. Tänu sellele saame juba "kulutatud" piletite summa abil mõnede osaliste variantide kohta varakult hinnata, et neist enam lahendust ei tule ning neile rohkem aega mitte kulutada. See lahendus mahub kõigis testides ajalimiidi piiridesse.

Testid

1. 6 linna, 3 piletit, vaid üks võimalik tee. 4 punkti.
2. 7 linna, 4 piletit, mitu võimalikku teed. 4 punkti.
3. 9 linna, 6 piletit. 4 punkti.
4. 8 linna, ühendused nagu servad kuubi tippude vahel, 0 piletit. 4 punkti.
5. 10 linna, 7 piletit, üks tasuta lend. 4 punkti.
6. 6 tipuga täisgraaf. 4 punkti.
7. 8 tipuga täisgraaf. 4 punkti.
8. 10 tipuga täisgraaf. 4 punkti.
9. 11 tipuga täisgraaf, kõik piletid makstakse kinni. 4 punkti.
10. 11 tipuga tihe graaf, 8 piletit. 4 punkti.

Kokku 40 punkti.